International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 4 Issue 12, December-2015

Mobile Cloud Computing System Using Self-Proxy Server

Soonhwa Sung[1], Jaecheol Ryou[2]

[1,2]Dept. of Computer science and Engineering College of Engineering Software Research Center (SOREC)
Chungnam National University, Yuseong-gu, Daejeon, 305-764, South Korea

ABSTRACT

Cloud computing has many advantages recently, but there are still many problems that need to outsource a collection of encrypted documents without revealing information about the contents of the documents efficiently. Besides, cloud computing application services are not provided practically due to drawback of applications. For these solutions, this scheme proposes a proxy server, which combines computing resources securely and distributes them at the appropriate season considering computational overhead. A distributed proxy server enables to outsource a collection of encrypted documents in parallel and classifies priority queue including keyword searches without revealing information about the contents of the documents and queries in the cloud.

Keywords: Cloud Computing, Self-Proxy Server, Priority Queuing, Parallel Search, Key Management.

INTRODUCTION

The cloud mobile ecosystem is vulnerable for the attackers because the digital assets and confidential data have some difficulty for cloud computing in secure way. The cloud system does no actual data processing on credit card transaction data in banking area. In the application provided by some company that runs a tax preparer algorithm, the users give out personal information, credit card information and the service provider runs some kind of algorithm to optimize the tax and finance strategy. If so, can we really store the bank account number and other balances to the cloud in a reliable and secure way? Could the users have the possibility to give the application a key by which it can download homomorphically encrypted data from the bank and run the proprietary algorithms over it and finally give to them a cipher text that only the initial user can decrypt and analyze?

The current generation of cloud computing infrastructures do not provide any security against untrusted cloud operators making them unsuitable for storing sensitive information. In cloud computing, if the users send data to an online storage service, the data can be encrypted before sending it. In this case, data privacy is assured and the provider can neither use nor analyze the data. However, analyzing current implementations of cloud computing services like Software-as-a-Service or even Infrastructure-as-a-Service, data can be encrypted during the transfer phase but it must arrive at destination as plaintext. The users and companies that use cloud computing services have to trust the service provider they choose. Techniques for encrypting a virtual machine and their attached disk volumes like [1] and [2] exist, but they only represent partial solutions. The main disadvantage of the solutions is that they require the decryption key to be transmitted at a given moment when booting up the virtual machine. The problem slows down the wide adoption of cloud services because of the lack of security for the most sensitive data.

Cloud Computing is an emerging economic and computing paradigm with the development of Internet technology. Various application service can be provided to satisfy users' requirements by the cloud computing [3]. One of cloud computing's constraints is that encrypted information cannot be processed within the cloud because currently there is no way to handle the data in a secure way once it is opened for computations. This is the field where homomorphic encryption should enter and fulfill the need for security and privacy. Cloud services are now used for different kinds of computing and business software, but current fully homomorphic encryption schemes are too slow to be considered practical [4].

This work was supported by the National Research Foundation of Korea (NRF) and the Center for Women In Science, Engineering and Technology (WISET) Grant funded by the Korean Government (Program for Returners into R&D by the Ministry of Science, ICT & Future Planning (MSIP)). This research was partly supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648). This work (Grants No. C0352986) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2015.

Cloud computing is a form of information technology which is being used where lesser investment in efficient software is needed. Cloud computing consists of access to applications and services is enabled over the network and it also require only access to internet connection. Possibly one can get access of the cloud with the use of an ordinary client simply anywhere and any-time and one needs a certain information facility, without any special software. Cloud computing also facilitates the clients for immediate access to pre-set common but valuable information resources that are eagerly available without a wide agreement making process. A kind of encryption that allows them to bypass some of maintaining the privacy of their requests is homomorphic encryption.

Like this, all usage scenarios of homomorphic encryption try to ensure privacy user data, but they have a drawback of applications. It is a computational overhead which performs operations on encrypted data. For this solution, various schemes have been studied in the recent years. The encryption scheme[5] supports addition of encrypted bits mod (eXclusive OR function). A number of encryption systems that are either additively or multiplicatively homomorphic followed the suit. Encryption systems of the scheme [6] , the Paillier encryption scheme [7,8] and its generation, a host od lattice-ased encryption schemes and others evolved [9,10,11]. A system used in involved additive and multiplicative encrypted texts which has more number of additions and just one multiplication. Constructing an encryption scheme that is both additively and multiplicatively homomorphic remained a major challenge [12]. The additive and multiplicative homomorphisms form a complete set of operations. The scheme enables performing any polynomial-time computation on encrypted data. Later, the scheme [12] constructed a fully homomorphic encryption which allows evaluation of arbitrary number of additions and multiplications on encrypted data [13,14].

However, these scheme have a drawback of applications including a computational overhead which performs operations on encrypted data without priority queuing. Therefore, this paper proposes that a distributed proxy server enables to outsource a collection of encrypted documents in parallel and classifies priority queue including keyword searches without revealing information about the contents of the documents and queries in the cloud to decrease a computational overhead. The rest of this paper is organized as follows: related work is presented in section 2, Self-Proxy Server (SPS) for secure and efficient mobile cloud computing is presented in section 3, analysis is detailed in section 4, and section 5 concludes the paper.

Related Works

Currently, outsourcing and managing large amount of data in cloud raise many challenges with respective to security. Among the various cryptographic encryption schemes, homomorphic schemes for a cloud computing are explained.

Fully Homomorphic Encryption:

We discuss what it means to be Fully Homomorphic Encryption (FHE). The previous scheme should allow one to compute arbitrary functions over encrypted data without the decryption key. The scheme consists of four algorithms such as Key Generation (KG), Message Encryption (ME), Message Decryption (MD) and Homomorphic Evaluation (HE).
Consider a booleancircuit f: $\{0, 1\}^h \rightarrow \{0, 1\}$.
h inputs $m_1, \ldots, m_h$ {0,1}', a pair of keys (pk, sk) andciphertextsc_i=Encrypt $_{pk}(m_i)$, i=1, h.
If Decrypt $_{sk}$(Evaluate (f, $c_1, \ldots, c_h$))=f ($m_1, \ldots, m_h$) [4].

Parallel Homomorphic Encryption:

We introduce Parallel Homomorphic Encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data through the use of an evaluation algorithm that can be efficiently executed in parallel. Using a PHE scheme, a client can outsource the evaluation of a function f on some private input x to a cluster of some machines. The client encrypts x and sends the cipher text and f to the controller. Using the cipher text, the controller generates n jobs that it distributes to the workers and the workers execute their jobs in parallel. When the entire computation is finished, the client receives a cipher text which it decrypts to recover f(x). The most immediate application of PHE is to the setting of outsourced computation where a weak computational device wishes to make use of the resources of a more powerful server. To be useful in this setting, it is crucial as follows: Running the encryption and decryption operations of the PHE scheme take less time than evaluating f on the input x directly.

The PHE scheme is multi-use in the sense that the evaluations of several different functions can be done on a single cipher text. Most computations are not completely parallelizable and require some amount of communication between machines. The specifics of how the computation and communication between processors are organized lead to particular architectures, each having unique characteristics in terms of computational and communication complexity. It follows that an important consideration in the design of several architecture independent models of parallel computation. Building and maintaining large-scale clusters requires a considerable amount of effort and resources, so a recent trend in cluster-computing has been to make use of cloud infrastructures. In certain setting, the cost is dominated

by the amount of work that is outsourced when the function being evaluated is very complex, or the client wishes to evaluate many different functions over its data [15].

In the work [16], the MapReduce model of computation has been formalized and studied. The work introduces a new complexity class that captures the power of the model and relates it to known models of parallel computation. It also presents new MapReduce algorithms for frequency counts, undirected s-t connectivity and for computing the minimum spanning tree of a dense graph. Other MapReduce algorithms have been proposed for a multitude of tasks, clustering high-dimensional data [17], processing large-scale graphs [18,19,20], and natural language processing [21].

Parallel Searchable Symmetric Encryption:

There are two high-level approaches to designing reasonably efficient and secure Searchable Symmetric Encryption schemes. The first approaches[22] associate to each document (an encrypted data structure) that can be tested for the occurrence of a given keyword. This approach naturally results in schemes with search time that is linear in n, where n the number of documents in the collection. The second approach [23] associates an encrypted inverted index to the entire document collection. This approach yields very efficient schemes since search time O(r), where r is the number of files that contain the keyword. Note that, O(r) is not only sub-linear, it is optimal. Due to its efficiency, the inverted index approach has been used in many subsequent works, including [24, 25, 26, 27].

Searchable Symmetric Encryption [28, 29, 30, 31, 32, 33] consists of three operations. Encryption transforms a keyword or file using a secret key into a ciphertext. Using the secret key, one can generate a search token for a specific keyword. Using this token, one can search in a set of ciphertexts for those that match the keyword. Thus, one can encrypt, but still search without decryption. The advantage compared to standard encryption is that the cloud can perform the search operation without the key and only return a matching subset for a query. Thus, the client does not have to download the entire data set and search himself.

Self-Proxy Server (SPS) for Secure and Efficient Mobile Cloud Computing

A cloud is basically a large scale distributed system where a data owner's data is replicated over multiple servers for high availability. The new paradigm of cloud computing provides an array of benefits and advantages over the previous computing paradigms and many organizations are migrating and adopting it. However, there are still a number of challenges because they are preventing the mobile users to take on cloud services.

OVERVIEW

One of security disadvantages in cloud computing is compromised servers. In a cloud computing situation, users do not have a choice of using physical acquisition toolkit. In that situation, where a server is compromised , they need to close their servers down till they get an earlier backup these security audits, then it leads to a noticeable decrease in customer trust. Cloud storage and cloud computing platforms were developed for cloud services and the users have the ability to outsource storage and computations on their data. In addition, it allows businesses to offload the task of maintaining a data-center. If the encryption scheme uses a homomorphic encryption, the cloud can still perform meaningful computations on the encrypted data. Cloud service providers make sure the data security in ordinary and man-made disasters. Generally, data is virtual across multiple sites. However, in the case of any such unnecessary event, a provider must do a comprehensive and quick restoration [34].

A model for key distribution based on data re-encryption is applied to a cloud computing system to address the demands of a mobile device environment, including limitations on mobile data usage, storage capacity, processing power, and battery etc. When an encrypted data is stored and decryption key is allocated to user, they can access data from cloud. However, what is the case when particular user is revoked? While a user is revoked and he has decryption key he can access data still, thus to overcome from this problem here is a need of immediate re-encryption of data by data owner. When re-encryption is done the newly generated, decryption keys are distributed to authorized users. This resolution will lead to performance bottleneck, particularly when there are many user revocations [35].

A solution is to apply a distributed self-proxy re-encryption technique, so this scheme proposes Self-Proxy Server (SPS). It coordinates and chooses keys by Key Manager (KM) whenever group membership changes. The distributed SPS provides not only encryption and decryption keys but also immediate re-encryption keys for shared data. After communicating with KM, it automatically receives necessary keys from KM by self-created algorithm. A distributed SPS scheme is one solution where multiple proxy are automatically deployed in several clouds. In **Table 1**, Mobile Cloud Provider (MCP) has significant resources and expertise in building and managing distributed cloud storage servers and computational services to data, owns and operates live cloud computing systems. Data Owner (DO) has data to be stored in the cloud and rely on the cloud for data computation, consists of both individual consumers and organizations. Key Manager (KM) generates and manages all data encryption, decryption and re-encryption keys. It is provided live cloud computing by MCP and governed by Trusted Third Party (TTP).

Table 1. Notation

Symbol	Descryption
MCP	Mobile Cloud Provider
DO	Data Owner
KM	Key Manager
TTP	Trusted Third Party
ACL	Access Control List
SPS	Self-Proxy Server
P_u	Public key for a user
S_u	Private key for a user
U_g	User group
P	Data Partition

We assume that a cloud operates by pre-defined protocols and policies between end users and cloud services and a key manager has basic capabilities on generating and managing different type of keys.

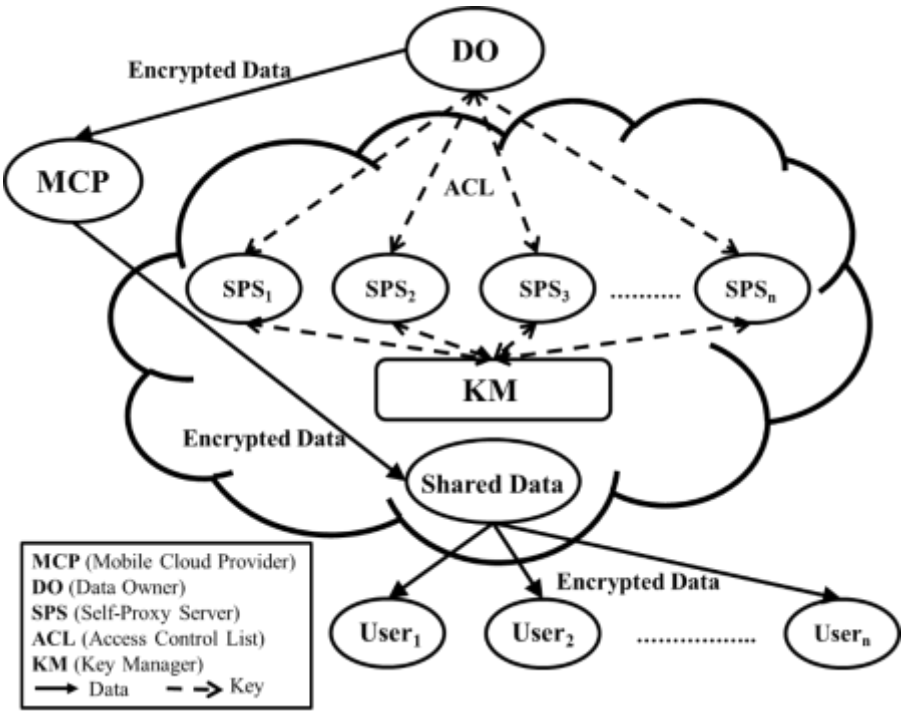

Figure1. A Distributed Mobile Cloud Computing Model with SPS

As shown in Fig.1,data owner of MCP shares data to many other cloud users. The data is encrypted with a key from KM, and then stored in the cloud along with Access Control List (ACL) indicating the user group. Upon access request from a user, the cloud communicates with SPS, based on ACL, and SPS requests a self-created algorithm. According to the self-created algorithm, SPS uses re-encryption algorithm to transfer the encrypted format that can be decrypted by the user's private key. The user can download the encrypted data from the cloud and use the decryption key.

PRIORITY QUEUING

Besides of classical virtualization, cloud computing uses in addition the capabilities of automation of services and multi-tenancy of users at common information resources. Common use of the same technological resources is the central feature of cloud computing. Traditionally, to control the dissemination of privacy-sensitive data, users establish a trusted server to store data locally in clear, and the server checks whether users are legitimate before accessing the data. However, this access control architecture is no longer applicable because data users and cloud servers are not in

the same trusted domain. This paper considers how to realize an efficient data access control design that leverages the cloud's computation resource richness when a large number of on-demand users desire a fine data ACL. To decrease a computational overhead, our scheme would like to classify the contents of the documents and queries in the cloud using priority queuing model in Fig. 2.

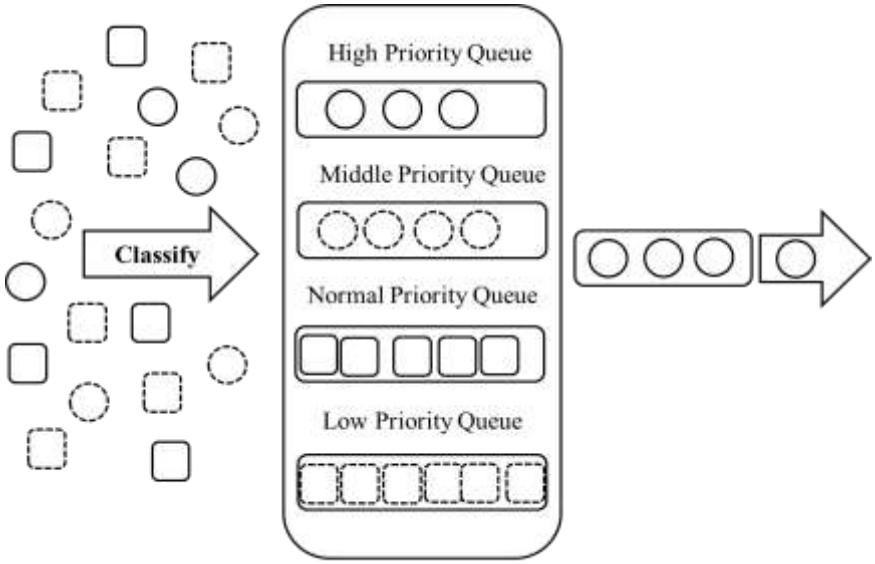

Figure 2. Priority Queuing Processing

Priority Queuing is small modification to the First In First Out (FIFO) queue where each packet is marked with a priority level. The priority level can be based on different attributes of the network flow including the source ports, destination ports, or application content of the packets. The packets are marked with Type of Service (ToS) indicator that is used by the scheme to determine the priority level of that particular packet. There are different queues created for each priority level and packets are placed in the appropriate queue and released from the queue in a FIFO manner. All of the packets in the higher priority queue for packets are processed before those in the lower queues such as in Fig. 2.

The process checks the high priority queue for packets, and if they exist, it sends them, until the queue is exhausted. This scheme has very little overhead on the router so it is relatively quick, however, if the amount of high priority data flows becomes excessive, the lower queues can start seeing delays in service or dropped packets due to overflowing queues. This issue is commonly referred as starvation. In addition, if high priority flow is misbehaving, the flows sharing the same queue can experience longer delays and increase jitter causing reduced Quality of Service (QoS). Thus, the proposed a distributed proxy server controls the data flows by his own exertions under the fixed limitations if the amount of high priority data flows becomes excessive.

A Distributed Cloud Design Using Key Management

To protect cloud data, we propose distributed SPS which interacts with KM whenever a mobile user requires cloud services. If the data in one of the cloud is corrupted, then the same data available in the other cloud could be accessed. Here the same data in different clouds would operate different keys from SPS which is under the control of KM. If same data sharing clouds work naturally, then users can choose SPS to re-encrypt the data depending on the physical location. It can be applied to reduce the computation cost for the data owner. KM generates public (P_u) and private (S_u) keys for each user belonging to the system and is responsible for maintaining an ACL for enforcing the authorized user set. A data partition P in the cloud is accessible by a user group U_g and belongs to the entire set of partitions. After confirming the access of user group U_g to data partition P, SPS interacts with KM.

Even if they download it directly from the cloud, MCP and other users cannot decode the data with or without authentication. After all read requests are initiated by users, this model is normally serviced through SPS which communicates with MCP. After MCP decides whether the data should be accessible by a user based on its ACL, it sends the information to SPS. SPS receives it and communicates with KM whether the service for a mobile user is offered. A distributed cloud model using key management is shown in Fig. 3.Previous model of key management in the cloud has a controller and a manager, but the proposed model has SPS instead of a controller and a manager.

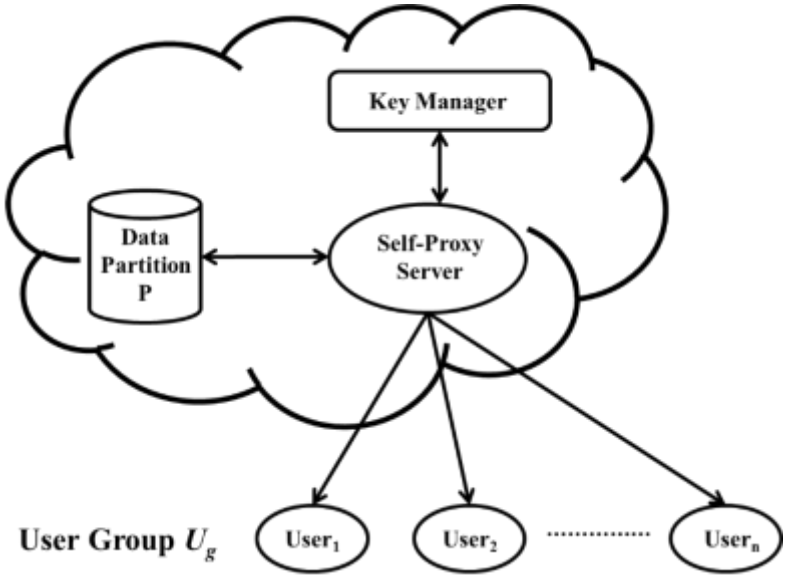

Figure 3. A Distributed Mobile Cloud Computing Model using Key Management

Cloud service providers make sure the data security and must do a comprehensive and quick restoration, but they concurrently do not serve these satisfaction in cloud computing. Therefore, we propose Self-Proxy Server (SPS) which should copy the data by communicating with KM, first of all considering priority policy when DO want to choose secure data. It automatically provides the data copy whenever the servers are shut down because of various reasons.

A Mobile Cloud Computing Model Using Parallel Searchable Keywords

We use keyword hash tables [36] to store some specific information for each one of the m keywords. The entries of the hash table ∂ are tuples (key, value), where the key is from a domain of exponential size $\{0,1\}^k$ and value is an encryption of a boolean value. However, the maximum number of entries in the table will be polynomial in k and equal to m, the number of keywords. If the key field is from $\{0,1\}^k$, and there are at most m entries in ∂ , then we say ∂ is a (k, m) hash table. For each $x \in \{0,1\}^k$,we denote with$\partial[x]$ the value associated with key x, if key x exists.

Searchable symmetric encryption allows a client to encrypt data so that it can later generate search tokens which the server can use to search over the encrypted data and return the appropriate encrypted files. The encryption algorithm takes as input an index δ, a sequence of n files $\mathbf{f} = (f_{i1}, \ldots, f_{in})$ that have unique identifiers $\mathbf{i}=(i_1, \ldots, i_n)$, and a universe of keywords $\mathbf{w}=(w_1, \ldots, w_m)$. File f_i has identifier iin order not to overload the notation. The index δ efficiently maps a keyword $w \in \mathbf{w}$ to set of identifies $\mathbf{i}_w \subseteq \mathbf{i}$ that correspond to a set of files $\mathbf{f}_w \subseteq \mathbf{f}$. The encryption algorithm outputs an encrypted index γ and a sequence of nciphertexts$\mathbf{c}=(c_{i1}, \ldots, c_{in})$, corresponding to the identifiers $\mathbf{i}=(i_1, \ldots, i_n)$. We assume all the ciphertexts include the identifiers of their plaintext files. The encrypted index γ and the ciphertexts **c** do not reveal any information about **f** other than the number of files n and their length, so they can be stored safely at an untrusted cloud provider [36].

Like this, the proposed scheme supports parallel searchable keywords and allows searchable symmetric encryption. SPS provides privacy as well as a comprehensive and quick restoration of cloud data. In addition, it manages private outsourced computations and an associated implementation for processing large datasets in preparation for chance failures of some machines or servers.

Fig. 4shows the overall flow of the execution overview in the proposed model. The invocations of cloud data are distributed across multiple machines by automatically partitioning the input data. The distributed input can be processed in parallel by different machines. The process using parallel searchable keywords is concurrently operated in a distributed SPS. The processing includes input files, map phase, intermediate files, reduce files and output files and is executed as follows:

1. DO forks data, and sends it to users and a distributed SPS.
2. SPS assigns map and reduce, where SPS interacts with KM.
3. User reads split file.
4. Set intermediate file after local write in map phase.
5. Remote read in reduce phase
6. SPS sends decryption key to remote user.
7. Get output file after write.

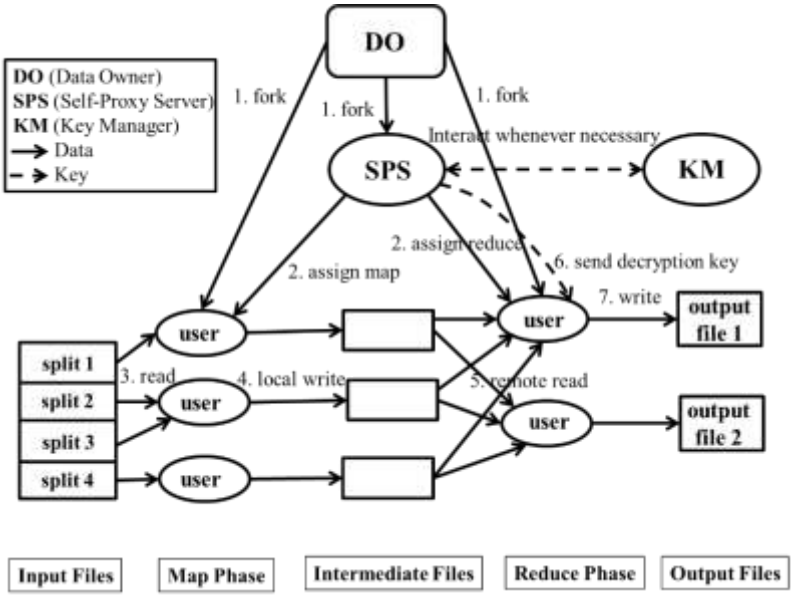

Figure 4. A Mobile Cloud Computing Model Using Parallel Search

After communicating SPS with KM, SPS decides the priority of searchable keywords using keyword hash tables by a priority policy. SPS chooses a higher priority queue and sends it to KM. After checking whether ACL is a higher priority queue for cloud data, KM generates public (P_u) and private (S_u) keys for each user. Processing in parallel for a distributed cloud data supports computations over encrypted data through the use of an evaluation algorithm that can be efficiently executed in parallel. The evaluation algorithm [36] enables a client to outsource a collection of encrypted documents in the cloud and retain the ability to perform keyword searches.

ANALYSIS

The proposed scheme is compared with another schemes [37, 38, 39] in a performance considering computational overhead. In comparison with attributed-based schemes, we can find out that the computation of these schemes increases when the number of attributes increases. The scheme is improved in computation cost because it supports priority queue with searchable parallel keywords instead of attributed-based. In Table 2, C denotes a pairing operation, E denotes an exponentiation group operation, M denotes a multiplication group operation, S denotes a signature operation, n denotes the number of attributes. For a user authentication, the scheme operates one time pairing operation and has three times in multiplication group operation for SPS and priority queuing of searchable keywords. It has three times in multiplication group operation for re-encryption including cloud computing confidentiality and has two times in multiplication group operation for re-decryption due to SPS.

Table 2. Performance Comparison

Item	Yu et al.[37]	Luo et al.[38]	Liang et al.[39]	Proposed Scheme
Encryption	(n+1)E+2M	(n+2)E+2M	(n+2)E+2M	C+3M
Decryption	-	(2n)C+3M	(n+2)C+2M	-
Re-encryption	(n)E	(2n+1)C+(n+1)M	(n+1)C+M	3M+1
Re-decryption	(n+1)C+2M	(2n+1)C+5M	(n+3)C+4M	2M+1

ACKNOWLEDGMENT

The authors appreciate Prof. Cheong Youn and Eunbae Kong for their helpful research supporting.

CONCLUSION

Currently, mobile cloud computing has a drawback of applications including a computational overhead which performs operations on encrypted data without priority queuing by searchable keywords. To solve the problem, this scheme suggests SPS for secure and efficient mobile cloud computing. SPS supports cloud computing applications which

classify a priority queue including searchable keywords. It interacts with KM, which acts efficient key management instead of a key manager and controller in cloud. The previous schemes support only parallel searchable keywords, meanwhile this scheme operates a priority queuing before searching parallel keywords. In addition, the previous attribute-based schemes have expensive computations. Therefore, the scheme can decrease many keys using in cloud computing application services and facilitate more rapid cloud computing growth because a distributed SPS interacts with KM.

REFERENCES

[1] [Online] Available: http://windows.microsoft.com/en-US/windows7/products/features/bitlocker
[2] [Online] Available: http://www.saout.de/misc/dm-crypt
[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G, Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing", Communications of the ACM, vol.53, pp.50-58, 2010.
[4] D.S. Maimut, A. Patrascu and E. Simion, "Homomorphic Encryption Scheme and Applications for a Secure Digital World", Journal of Mobile, Embedded and Distributed Systems, vol. IV, no. 4, 2012.
[5] N. Yukun, T. Xiaobin, C. Shi, W. Haifeng, Y. Kai and B. Zhiyong, "A Security Privacy Protection Scheme for datacollection of Smart Meters Based on Homomorphic Encryption", EUROCON, pp.1401-1405, 2013.
[6] A. Peter, E. Tews and S. Katzenbeisser, "Efficiently Outsourcing MultipartyComputation Under Multiple Keys", IEEE Transactions on Information Forensics and Security, vol.8, no.12, pp.2046-2058, 2013.
[7] N. Saputro and K. Akkaya, "Performance Evaluation of Smart Grid Data Aggregation via Homomorphic Encryption", Wireless Communications and Networking Conference (WCNC), pp.2945-2950, 2012.
[8] X. Chen and Q. Huang, "The Data Protection of Map-Reduce Using Homomorphic Encryption", 4th IEEE International Conference on Software Engineering Service Science (ICSESS), pp.419-421, 2013.
[9] L. Chen, Z. Tong, W. Liu and C. Gao, "Non-meteractive Exponential Homomorphic Encryption Algorithm", International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp.224-227, 2012.
[10] D. J. Guan, E.S. Chen-Yu and T. Zhuang, "Detect Zero by Using Symmetric Homomorphic Encryption", 8th Asia Joint Conference on Information Security (AsiaJCIS), pp.1-7, 2013.
[11] J. Li, S. Chen and D. Song, "Security Structure of Cloud Storage Based on Homomorphic Encryption Scheme", 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), pp.224-227, 2012.
[12] C. Gentry and S. Halevi, "Fully Homomorphic Encryption Without Squashing Using Depth-3 Arithmetic Circuits", 52nd Annual Symposium on Foundations of Computer Science (FOCS), pp.107-109, 2011.
[13] P. Zhu and G. Xiang, "The Protection Method for Mobile Code Based on Homomorphic Encryption and Data Confusion", 5th International Conference on Management of e-Commerce and e-Government (ICMeCG), pp.256-260, 2011.
[14] [14]F. Jin, Y. Zhu and X. Luo, "Verifiable Fully Homomorphic Encryption Scheme", 2nd International Conference on Consumer Electronics Communications and Networks (CECNet), pp.743-746, 2012.
[15] [15]S. Kamara and M. Raykova, "Parallel Homomorphic Encryption",[Online] Available: http://research. microsoft. com/pubs/155853/phe.pdf
[16] H. Karloff, S. Suri and S. Vassilvitskii, "A Model of Computation for Mapreduce", In Symposium on Discrete Argorithms(SODA'10), SIAM, pp.938-948, 2010.
[17] A. Das, M. Datar, A. Garg and S. Rajaram, "Google News Personalization: Scalable Online Collaborative Filtering", In Conference on World Wide Web(WWW'07), ACM, New York, NY, USA, pp.271-280, 2007.
[18] F. Chierichetti, R. Kumar and A. Tomkins, "Max-Cover in Map-Reduce", In Conference on World Wide Web, ACM, New York, NY, USA, pp.231-240, 2010.
[19] U. Kang, C. E. Tsourakakis and C. Faloutsos, "Pegasus: A Peta-Scale Graph Mining System Implementation and Observations", In International Conference on Data Mining(ICDM'09), IEEE Computer Society, Washington, DC, USA, pp.229-238, 2009.
[20] J. Lin and M. Schatz, "Design Patterns for Efficient Graph Algorithm in Mapreduce", In Workshop on Mining and Learning with Graphs, ACM, New York, NY, USA, pp.78-85, 2010.
[21] D. Rao and D. Yarowsky, "Ranking and Semi-Supervised Classification on Large Scale Graphs Using Map-Reduce", In Workshop on Graph-based Methods for Natural Language Processing, Association for Computational Linguistics, Morristown, NJ, USA, pp.58-65, 2009.
[22] Y. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data", In Applied Cryptography and Network Security(ACNS), pp.442-455, 2005
[23] R. Curtmola, J. Garary, S. Kamara and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions", In Computer and Communications Security (CCS), pp.79-88, 2006.
[24] M. Chase and S. Kamara, "Structured Encryption and Controlled Disclosure", In Theory and Application of Cryptology and Information Security(ASIACRYPT), pp.577-594, 2010.
[25] S. Kamara, C. Papamanthou and T. Roeder, "Dynamic Searchable Symmetric Encryption", In Computer and Communications Security(CCS), pp.965-976, 2012.
[26] K. Kurosawa and Y. Ohtaki, "UC-Secure Searchable Symmetric Encryption", In Financial Cryptography(FC), pp.285-298, 2012.
[27] P. van Liesdonk, S. Sedghi, J. Doumen, P. H. Hartel and W. Jonker, "Computationally Efficient Searchable Symmetric Encryption", In Secure Data Management(DSM), pp.87-100, 2010.
[28] D.Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation", In Proceedings of the 21st Network and Distributed System security Symposium, NDSS, 2014.
[29] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.C. Rosu, and M. Steiner, "Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries", In Proceedings of the 33rd Cryptology Conference, CRYPTO, 2013.
[30] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and

Efficient Constructions", Journal of Computer Security, 19(3), 2011.

[31] S. Kamara and C. Papamanthou, "Parallel and Dynamic Searchable Symmetric Encryption", In Proceedings of the 17th International Conference on Financial Cryptography and Data Security, FC, 2013.

[32] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic Searchable Symmetric Encryption", In Proceedings of the 19th ACM Conference on Computer and Communications Security, CCS, 2012.

[33] M. Naveed, M. Prabhakaran, and C. Gunter, "Dynamic searchable Encryption via Blind Storage", In Proceedings of the 35th IEEE Symposium on Security and Privacy, S&P, 2014.

[34] MayankPatwal and Tanushri Mittal, "A Survey of Cryptographic Based Security Algorithms for Cloud Computing", HCTL Open International Journal of Technology Innovation and Research(IJTIR), vol.8, March 2014.

[35] Mrs. Pooja A. Uplenchwar and Mrs. L. H. Patil, "Data Security in Unreliable Cloud Using Access Control and Access Time", International Journal of Scientific & Engineering Research, vol. 4, Issue 12, Dec. 2013.

[36] S. Kamara and C. Papamanthou,"Parallel and Dynamic Searchable Symmetric Encryption", In Proceedings of the 19th ACM Conference on Computer and Communications Security, CCS, 2012.

[37] S. Yu, C. Wang, K. Ren and W. Lou, "Attribute-based Data Sharing with Attribute Revocation", Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp.261-270, 2010.

[38] S. Luo, J. Hu and Z. Chen, "Ciphertext Policy Attribute-based Proxy Re-encryption", Information and Communications Security, pp.401-415, 2010.

[39] X. Liang, Z. Cao, H. Lin and J. Shao, "Attribute-based Proxy Re-encryption with Delegation Capabilities", Proceedings of the 4th International Symposium on information, Computer, and Communications Security, pp.276-286, 2009.